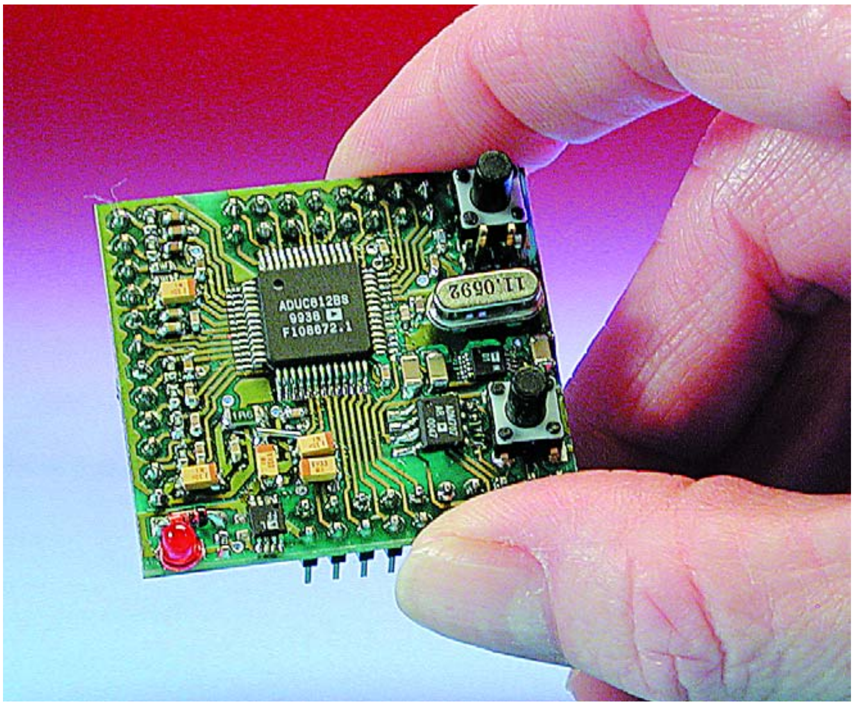
ISAC (4)

part 4: basic applications

By Prof. B. vom Berg, P. Groppe and M. Müller-Aulmann

In this fourth instalment of the series we will look at our first applications of the ISAC cube and the simple motherboard, which will give a good introduction to the possibilities of the device. Complete example programs developed for these applications, along with extra information, are available for download from the *Elektor Electronics* website.



Before we present the individual example applications, a few initial remarks are in order regarding the example programs and their use.

- The applications and demonstration programs are written in C51 code in such a way that (with one exception) they can be used with the restricted version of μVision2 (maximum code size of 2 Kbytes, no floating-point arithmetic). For larger application programs the full version of μVision2, or another 8051 tool, must be resorted to.
- In order that something 'visible' is produced, all the programs guide the user via, and produce output on, the PC's monitor. That means that you will need to run a terminal program on your PC. We have assumed that you will be using HyperTerminal, since this program is included with Windows. You can, however, use any other terminal program: the required communications parameters are 9600 baud, 8 data bits, no parity, 1 stop bit.
- Before starting a program on the ISAC cube, you should start up your chosen terminal program and then press the reset button on the cube.
- **Very important:** if you have been using HyperTerminal and then want to download a new program into the ISAC cube using the serial downloader, you must first clear the existing connection with HyperTerminal (using 'Disconnect' on HyperTerminal). Otherwise HyperTerminal will keep control of the COM port and the serial downloader will not be able to access it. An apparently meaningless error message will appear if the serial downloader is unable to find the ADuC812. In this case you will have to cancel the error message, release the connection in HyperTerminal, and then relaunch the downloader.
- Communication between the microcontroller core and the on-chip peripherals uses the 8051-style Special Function Register (SFR) (**Figure 1**). Further information on the meaning of the individual bits within the SFR can be found in the ADuC812 data sheet or in specialist 8051 books.

D/A converter application

Outputting different signals: With the aid of program **dac_1.c** two different signals (a sawtooth and a squarewave) can be output using the D/A converter at two different resolutions (8 bits and 12 bits). Connect an oscilloscope to the relevant outputs to verify the results.

The reference voltage used here is the digital power supply voltage V_{DD} (=+5 V), and so the output voltages produced lie in the range 0 to +5 V.

A/D converter application

Reading the voltages on the input channels:

On starting up program **adc_1.c** you will be asked to supply the number of the channel to be read, where you have the following possibilities:

- number 0-7: external measurement channels ADC 0 to ADC 7.
- number 8: channel connected to the on-chip temperature sensor (see next example).

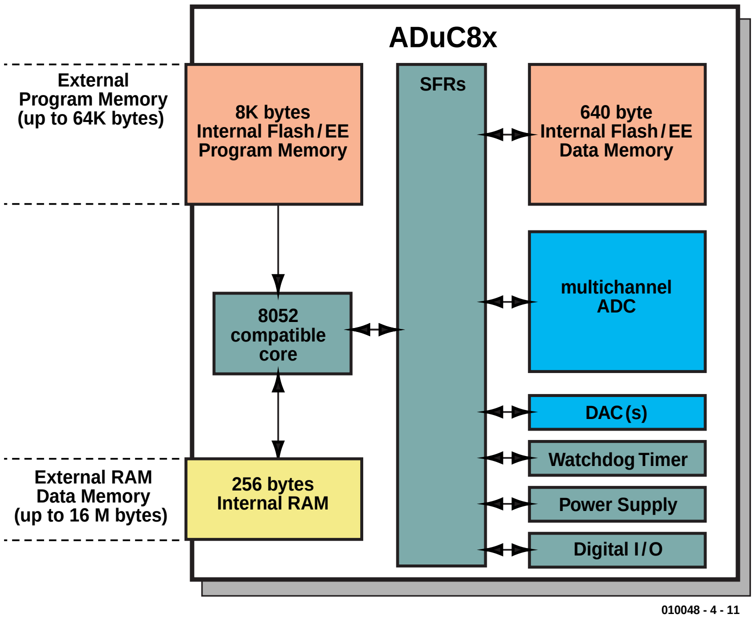


Figure 1. The SFR register.

When you have entered a number, the voltage present at the selected input channel is converted once per second and the result displayed as a decimal and as a hexadecimal value on the PC's monitor. Readings are stopped by pressing the reset button, and a new input channel can then be chosen.

Note: The A/D converter has a resolution of 12 bits and can operate with two different reference voltages:

1. Using the internal 2.5 V reference voltage. In this case 1 LSB corresponds to a voltage step of 610 µV.
2. Using an external reference voltage in the range +2.3 V to +5.0 V, connected to the V_{Ref} pin. In view of the 12 bit resolution, this voltage should be specially filtered and regulated. If, for example, V_{Ref} is connected to AV_{DD} supply (+5 V), 1 LSB will correspond to a voltage step of 1.2 mV.

The input voltage to be measured must **always** lie between 0 V and the reference voltage used (whether internal or external), i.e. in the range 0 to +2.5 V when using the internal reference voltage or in the range 0 to +5 V if an external 5 V reference is used.

Measuring the internal die temperature and the ambient temperature:

The ADuC812 includes an on-chip temperature sensor that produces a temperature dependent voltage that is connected to the ninth channel (i.e., channel number 8) of the A/D converter. The datasheet gives the following information about this sensor: the output voltage is 600 mV at a temperature ϑ = 25°C, and the temperature dependence is –3 mV/°C. If the temperature rises by 1°C the temperature-dependent voltage will drop by 3 mV and vice versa. But there is a **very important point** to note here: these figures are only **typical values** and vary from device to device. If you compare ADuC812s from different production batches, you will find rather different values. Analog Devices do guarantee, however, that the temperature-dependent voltage is highly linear in die temperature.

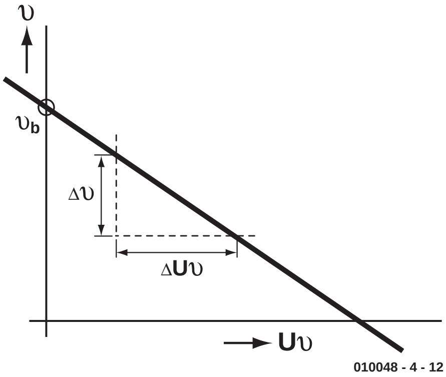


Figure 2. The linear relationship between voltage U_ϑ and temperature ϑ.

This relationship is characterised by a (linear) equation U_ϑ=f1(ϑ) (or ϑ=f2(U_ϑ)) (**Figure 2**). For the user, this means that two calibration readings have to be taken to establish the parameters of the linear equation before meaningful temperature values can then be obtained. If you then have an ADuC812 from the same production batch (which is usually the case if they come from the same tube), nothing needs to be changed. If, however, you buy ADuC812s from different batches, this calibration will have to be repeated several times over. In the worst case a different set of parameters for the linear relationship will have to be obtained for each

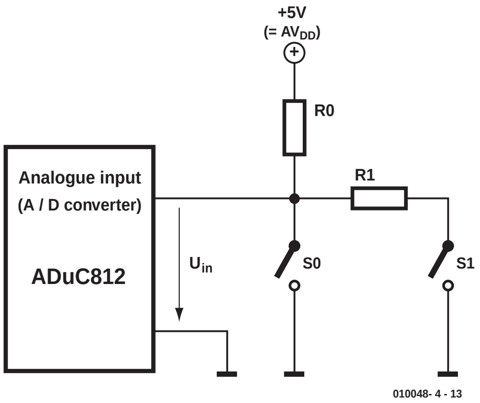


Figure 3. Voltage divider at the input to the A/D converter switched by push-buttons.

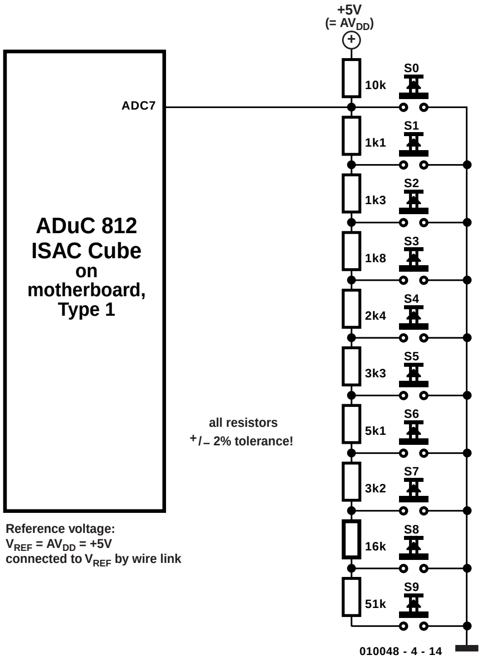


Figure 4. Series of push-buttons connected to an analogue input pin.

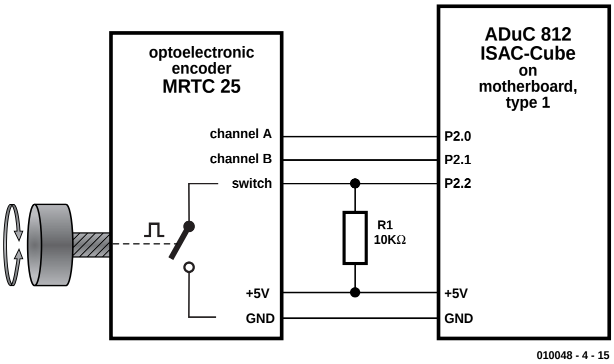


Figure 5. Connecting an opto-electronic shaft encoder.

Table 1

Characteristics of the MRTC25 opto-electronic shaft encoder

Electrical characteristics:

Resolution (pulses/360°)	25
Channels	2 (A and B; 90° phase offset)
Supply voltage	5 V DC ±0.5 V
Current consumption	20 mA
Output signals	TTL-level squarewave; internal 10 kΩ pull-up
Push-button	5 V, 10 mA (non-inductive load)
Operating life	> 1,000,000 cycles

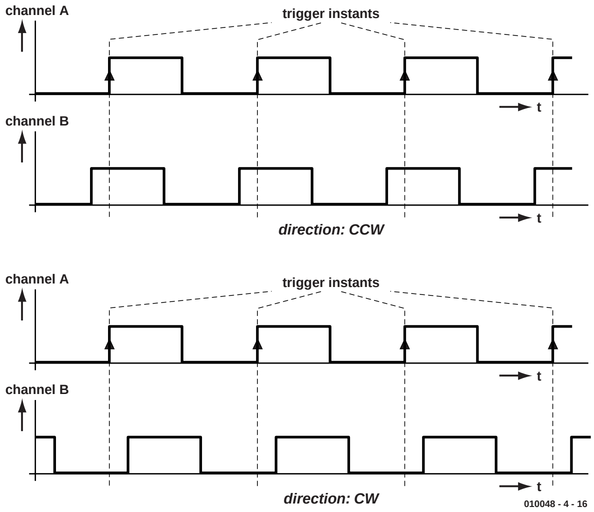


Figure 6. Timing diagram for the two directions of rotation.

ADuC812. The program **temp812.c** has been developed to help in the calculation. When the program is downloaded you will also find instructions on how to determine the parameters of the linear relationship, on **measuring the ambient temperature**, and on deriving the settings programmed into temp812.c. Observe the instructions given on how to use the program.

Push-buttons connected to an analogue input:

The basis for this method of connecting push-buttons, which allows an entire keyboard to be connected to one analogue input pin of the A/D converter, is a 'multiply switched voltage divider'. We start with a simple voltage divider (**Figure 3**).

With this circuit we have:

No button pressed: $U_{in}=5$ V
Button 0 pressed: $U_{in}=0$ V
Button 1 pressed: $U_{in}=5\ [R1/(R0+R1)]$ V

If the two resistors R1 and R0 have the same value, for example 1 kΩ, we have $U_{in}=2.5$ V.

For each button pressed, a different and distinguishable voltage appears on the analogue input and is converted by the A/D converter. In this way, for each button that might be pressed we obtain a different digital value and hence we can identify uniquely the button pressed. There is a further advantage to this circuit: if more than one button is pressed simultaneously, only the button nearest the analogue input will be recognised. In other words, we can build a keyboard in which the individual keys are prioritised. If, for example, button 1 is already pressed and subsequently button 0 is pressed, U_{in} will immediately be pulled to the voltage corresponding to button 0 (0 V). Thus button 1 is overridden and will no longer be recognised, while button 0 has the highest priority and will always be recognised. When button 0 is subsequently released, the other buttons can then be read.

In order to use a large number of buttons close-tolerance resistors are required, as well as a well-regulated power supply to the voltage divider. **Figure 4** shows a practical circuit for a keyboard with ten push buttons.

A small disadvantage of this idea

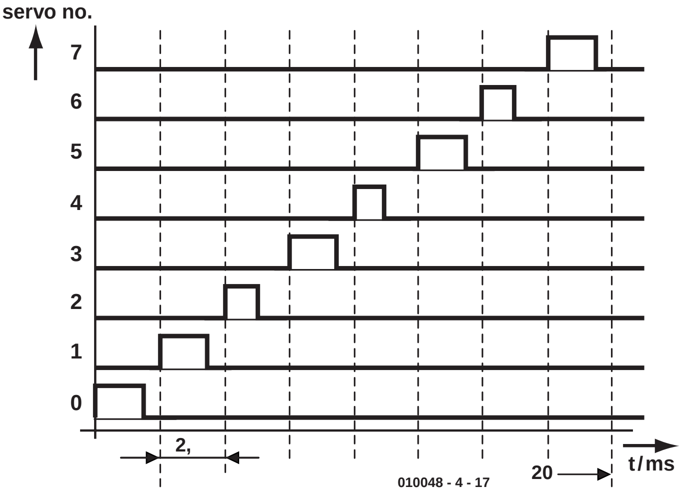


Figure 7. Timing diagram for controlling a servo.

should not be ignored: the buttons can of course only be read while they are pressed: there is no automatic keyboard buffer. This omission can, however, be rectified by dedicated software. A suitable program for reading a 10-key keyboard connected to analogue input pin ADC7 of the ADuC812 is provided in program **ana_tast.c**. When downloading this program you will also find a table showing the relationship between the button pressed, the input voltage U_{in} and the converted digital value, safe push button reading interval and button priority level (0=highest priority).

Digital I/O port application

Use of an opto-electronic shaft encoder:
An opto-electronic shaft encoder makes a simple and elegant input device. Here we use a type MRTC25 from Megatron, which offers two output channels (A and B) for count and direction indication (clockwise/counterclockwise). It also provides a contact that closes when the shaft is pressed down.

Connection to the ISAC cube or to the simple (type 1) motherboard proceeds as shown in **Figure 5**. The timing diagram for clockwise and counterclockwise rotation is shown in **Figure 6**. Electrical characteristics of the shaft encoder are given in **Table 1**. Program **drehge_1.c** shows how to interface to this interesting input device.

Control of eight servos:
Using the eight digital outputs of port 2 and a little software, in this case written in assembler, it is easy to control eight servos. For each servo a control pulse of between 0.9 ms and 2.1 ms must be generated every 20 ms (**Figure 7**). The simplest approach is to divide the 20 ms period into eight subintervals and then drive each servo in turn from the appropriate output on port 2 to the desired position. Program **servo.asm** generates the 2.5 ms intervals using Timer 2 in auto-reload mode and generates the required servo pulse durations using Timer 0 as a 16 bit timer.

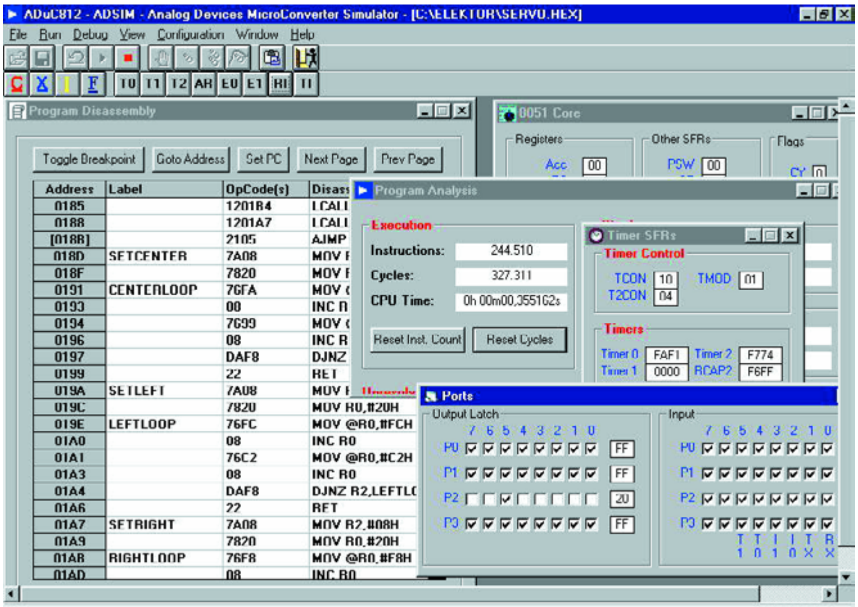
Interval: 20 ms / 8 = 2.5 ms (Timer 2)
Servo pulse width: 0.9 ms to 2.1 ms (Timer 0)

In the Timer 2 interrupt service routine Timer 0 is started with the pulse width required for the servo, and the appropriate pin of port 2 is set. In the Timer 0 interrupt service routine all port 2 pins are cleared and the timer stopped.
In order to achieve exact timing in the main program, an extra software delay is employed in the Timer 2 interrupt service routine. This short program is readily tested using the ADuC812 simulator (**Figure 8**).

Of course, applications for the ADuC812 are limited only by your imagination. With a little effort on the software side up to 24 servos could be controlled using the 24 digital outputs. When used in combination with the I^{2}C interface we can even construct a replacement 'I^{2}C Servo Interface' (*Elektor Electronics*, September 2001) — and our replacement offers a resolution of more than 1000 steps.

The servos could also be controlled by sensors via the eight analogue inputs — which is exactly what this series (ISAC — Intelligent Sensor/Actuator Controller) is about.

(010048-4)



Address	Label	OpCode(s)	Disass
0185		1201B4	LCALL
0188		1201A7	LCALL
[018B]		2105	AJMP
018D	SETCENTER	7A08	MOV R
018F		7820	MOV R
0191	CENTERLOOP	76FA	MOV (
0193		00	INC R
0194		7699	MOV (
0196		08	INC R
0197		DAF8	DJNZ
0199		22	RET
019A	SETLEFT	7A08	MOV R
019C		7820	MOV R0,#20H
019E	LEFTLOOP	76FC	MOV @R0,#FCH
01A0		08	INC R0
01A1		76C2	MOV @R0,#C2H
01A3		08	INC R0
01A4		DAF8	DJNZ R2,LEFTLC
01A6		22	RET
01A7	SETRIGHT	7A08	MOV R2,#08H
01A9		7820	MOV R0,#20H
01AB	RIGHTLOOP	76F8	MOV @R0,#F8H
01AD		08	INC R0

Acc 00
PSW 00
CY 0

Instructions: 244.510
Cycles: 327.311
CPU Time: 0h 00m00,355162s

TCON 10
TMOD 01
T2CON 04

Timer 0 FAF1
Timer 2 F774
Timer 1 0000
RCAP2 F6FF

	7	6	5	4	3	2	1	0	
P0	☑	☑	☑	☑	☑	☑	☑	☑	FF
P1	☑	☑	☑	☑	☑	☑	☑	☑	FF
P2	☐	☐	☑	☐	☐	☐	☐	☐	20
P3	☑	☑	☑	☑	☑	☑	☑	☑	FF

	7	6	5	4	3	2	1	0
P0	☑	☑	☑	☑	☑	☑	☑	☑
P1	☑	☑	☑	☑	☑	☑	☑	☑
P2	☑	☑	☑	☑	☑	☑	☑	☑
P3	☑	☑	☑	☑	☑	☑	☑	☑
			T1	T0	I1	I0	TX	RX

Figure 8. servo.hex in the ADuC812 simulator (pulses output for servo 5 on pin P2.5).

Component source:
MEGATRON Bauelemente,
Hermann-Oberth-Strasse 7,
D-85640 Putzbrunn, Munich, Germany.
Tel.: +49 89 46094 146